МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ  
РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

ПЕРМСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ  
ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Кафедра информационные технологии и автоматизированные системы

**Дисциплина Информатика**

**Контейнеры: стек**

Выполнил студент ИВТ-22-2б:

Мифтахов Марат Ринатович

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Проверил доцент кафедры ИТАС:

Полякова Ольга Андреевна

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Пермь, 2023

**Постановка задачи**

1. Определить класс-контейнер.
2. Реализовать конструкторы, деструктор, операции ввода-вывода, операцию присваивания.
3. Перегрузить операции, указанные в варианте.
4. Реализовать класс-итератор. Реализовать с его помощью операции последовательного доступа.
5. Написать тестирующую программу, иллюстрирующую выполнение операций.

**Задание**

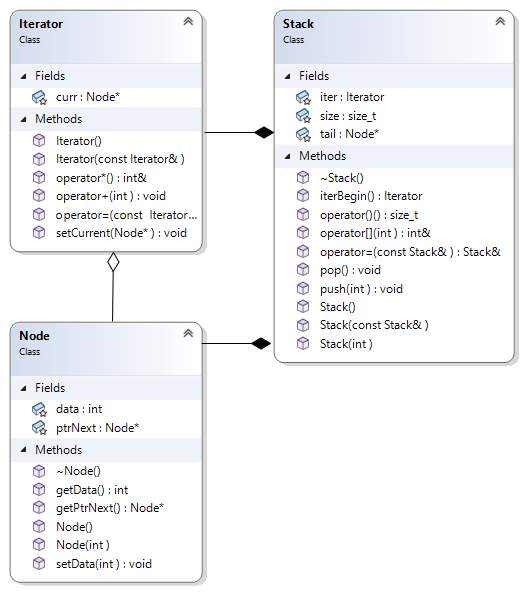
1. Класс-контейнер стек
2. Поля типа int

Операции:

1. [] - доступ по индексу
2. () - определение размера списка
3. вектор — уможение элементов списков a[i]b[i]
4. +n – переход вправо на кол-во элементов n (класс-итератор)

**Диаграмма классов**

Ниже представлена UML-диаграмма классов.



**Программный код**

Файл stack.h:

#include <iostream>

using namespace std;

#pragma once

class Stack;

class Node {

friend class Stack;

friend class Iterator;

protected:

Node\* ptrNext;

int data;

public:

Node();

Node(int);

~Node();

void setData(int);

int getData();

Node\* getPtrNext();

};

class Iterator {

friend class Stack;

protected:

Node\* curr;

public:

Iterator();

Iterator(const Iterator&);

void setCurrent(Node\*);

Iterator& operator=(const Iterator&);

int& operator\*();

void operator+(int);

};

class Stack

{

protected:

size\_t size;

Iterator iter;

Node\* tail;

public:

Stack();

Stack(int);

Stack(const Stack&);

~Stack();

void pop();

void push(int);

Iterator iterBegin();

Stack& operator=(const Stack&);

size\_t operator()();

int& operator[](int);

friend ostream& operator<<(ostream&, const Stack&);

friend istream& operator>>(istream&, Stack&);

friend Stack operator\*(Stack&, Stack&);

};

Файл list.cpp:

#include "stack.h"

Node::Node() {

data = 0;

ptrNext = nullptr;

}

Node::Node(int data) : data(data) {

ptrNext = nullptr;

}

Node::~Node() {

if (ptrNext != nullptr) delete ptrNext;

}

void Node::setData(int data) {

this->data = data;

}

int Node::getData() {

return data;

}

Node\* Node::getPtrNext() {

return ptrNext;

}

Stack::Stack() {

size = 0;

tail = nullptr;

iter.curr = nullptr;

}

Stack::Stack(int data) {

tail = new Node(data);

size = 1;

iter.curr = tail;

}

Stack::~Stack() {

tail->~Node();

}

Stack::Stack(const Stack& stack) {

size = 0;

if (tail != nullptr) tail->~Node();

Node\* stackPointer = stack.tail;

while (stackPointer != nullptr) {

int data = stackPointer->data;

this->push(data);

stackPointer = stackPointer->ptrNext;

}

iter.curr = tail;

}

void Stack::pop() {

if (tail != nullptr) {

Node\* tailPointer = tail;

if (tail->ptrNext != 0) tail = tail->ptrNext;

else tail = 0;

delete tailPointer;

size--;

}

iter.curr = tail;

}

void Stack::push(int data) {

if (tail == nullptr) {

tail = new Node(data);

}

else {

Node\* tailPrevPointer = tail;

tail = new Node(data);

tail->ptrNext = tailPrevPointer;

}

iter.curr = tail;

size++;

}

Iterator Stack::iterBegin() {

return iter;

}

Stack& Stack::operator=(const Stack& stack) {

size = 0;

if (tail != nullptr) tail->~Node();

Node\* stackPointer = stack.tail;

while (stackPointer != nullptr) {

int data = stackPointer->data;

this->push(data);

stackPointer = stackPointer->ptrNext;

}

iter.curr = tail;

return \*this;

}

size\_t Stack::operator()() {

return size;

}

int& Stack::operator[](int index) {

Node\* nodePointer = tail;

for (int i = 0; i < index && nodePointer != nullptr; i++) {

if (nodePointer->ptrNext != nullptr) nodePointer = nodePointer->ptrNext;

}

return nodePointer->data;

}

ostream& operator<<(ostream& out, const Stack& stack) {

Node\* nodePointer = stack.tail;

while (nodePointer != nullptr) {

out << nodePointer->getData() << '\n';

nodePointer = nodePointer->getPtrNext();

}

return out;

}

istream& operator>>(istream& in, Stack& stack) {

int data;

in >> data;

stack.push(data);

stack.iter.setCurrent(stack.tail);

return in;

}

Stack operator\*(Stack& fStack, Stack& sStack) {

Stack tmpStack;

int maxSize = fStack.size > sStack.size ? fStack.size : sStack.size;

int minSize = fStack.size <= sStack.size ? fStack.size : sStack.size;

for (int i = 0; i < minSize; i++) {

int a = fStack[i];

int b = sStack[i];

tmpStack.push(a \* b);

}

for (int i = 0; i < maxSize - minSize; i++) {

tmpStack.push(0);

}

return tmpStack;

}

Iterator::Iterator() {

curr = nullptr;

}

Iterator::Iterator(const Iterator& iter) {

curr = iter.curr;

}

void Iterator::setCurrent(Node\* node) {

curr = node;

}

int& Iterator::operator\*() {

return curr->data;

}

Iterator& Iterator::operator=(const Iterator& iter) {

curr = iter.curr;

return \*this;

}

void Iterator::operator+(int num) {

for (int i = 0; i < num && curr != nullptr; i++) {

if (curr->ptrNext != nullptr) curr = curr->ptrNext;

}

}

Файл main.cpp:

#include <iostream>

#include "stack.h"

using namespace std;

Stack createAndFillStack(size\_t);

int main() {

size\_t stackSize = 0;

cout << "Enter the stack size: "; cin >> stackSize;

cout << "Enter da elements: " << endl;

Stack stack1 = createAndFillStack(stackSize);

cout << "Enter the stack size: "; cin >> stackSize;

cout << "Enter da elements: " << endl;

Stack stack2 = createAndFillStack(stackSize);

Stack stack3 = stack1 \* stack2;

cout << "Mult: " << endl;

cout << stack3;

return 0;

}

Stack createAndFillStack(size\_t stackSize) {

Stack stack;

for (int i = 0; i < stackSize; i++) {

cin >> stack;

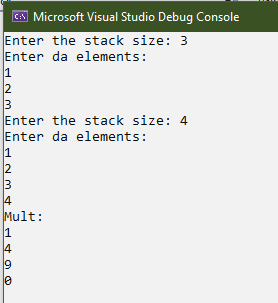
}

return stack;

}

**Вывод программы**

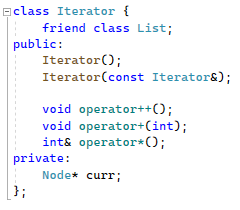
Ниже представлен вывод программы на консоль.

****

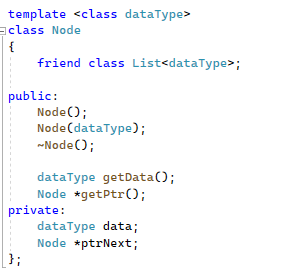
**Контрольные вопросы**

1. Что такое абстрактный тип данных? Привести примеры АТД.

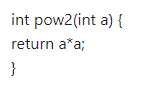
Ответ: АТД - тип данных, определяемый только через операции, которые могут выполняться над соответствующими объектами безотносительно к способу представления этих объектов.



1. Привести примеры абстракции через параметризацию.

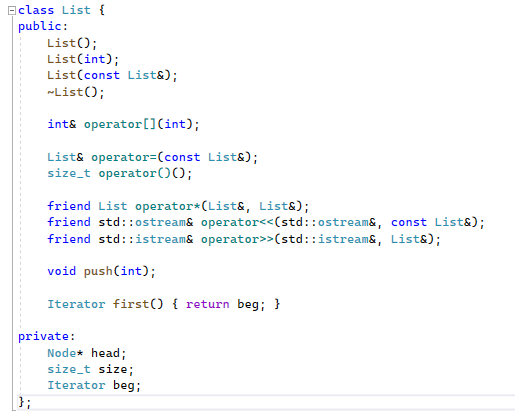
Ответ: класс Node – параметризованный класс

1. Привести примеры абстракции через спецификацию.

Ответ: функция возведения числа в квадрат

1. Что такое контейнер? Привести примеры.

Ответ: контейнер — набор однотипных элементов с операциями над ними.



1. Какие группы операций выделяют в контейнерах?

Ответ:

* + 1. Операции доступа к элементам
    2. Операции добавления и удаления элементов
    3. Операции поиска элементов
    4. Операции объединения контейнеров
    5. Специальные операции

1. Какие виды доступа к элементам контейнера существуют? Привести примеры.

Ответ: 1. через внутренние операции 2. через итератор (см. данную лабораторную)

1. Что такое итератор?

Ответ: итератор — объект, обеспечивающий доступ к элементам контейнера.

1. Каким образом может быть реализован итератор?

Ответ: итератор может быть реализован как класс

1. Каким образом можно организовать объединение контейнеров?

Ответ: сцепление контейнеров, объединение упорядоченных контейнеров (с упорядоченным полученным); объединение контейнеров, как пересечения/ объединения множеств

1. Какой доступ к элементам предоставляет контейнер, состоящий из элементов «ключ-значение»?

Ответ: ассоциативный доступ по ключу

1. Как называется контейнер, в котором вставка и удаление элементов выполняется на одном конце контейнера?

Ответ: стек

12.Какой из объектов (a,b,c,d) является контейнером?

* + 1. int mas=10;
    2. 2. int mas;
    3. 3. struct {char name[30]; int age;} mas;
    4. 4. int mas[100];

Ответ: D

13.Какой из объектов (a,b,c,d) не является контейнером?

* + 1. int a[]={1,2,3,4,5};
    2. int mas[30];
    3. struct {char name[30]; int age;} mas[30];
    4. int mas;

Ответ: h

14. Контейнер реализован как динамический массив, в нем определена операция доступ по индексу. Каким будет доступ к элементам контейнера?

Ответ: прямой доступ через перемещение указателя.

15. Контейнер реализован как линейный список. Каким будет доступ к элементам контейнера?

Ответ: Последовательный доступ: нужно пройтись по предыдущим элементам, чтобы достичь заданного.